Article

On Energy Optimization in Cryptographic Hash Functions and Symmetric Key Protocols †

Swapnoneel Roy, Priyanka D. Harish and S. Raghu Talluri

School of Computing, University of North Florida, Jacksonville, FL 32224, USA
* Correspondence: s.roy@unf.edu; Tel.: +1-904-620-2182

§ This paper is an extended version of our paper published in IEEE GreenCom 2014.

Abstract: In this work we study the energy consumption by various modern secured hash functions (MD2, MD5, SHA-1, and SHA-2) and modern symmetric key encryption protocols (Blowfish, DES, 3DES, and AES) from the *algorithmic* perspective. We identify various parameters that moderate energy consumption of these hashes and protocols. Our work is directed towards redesigning or modifying these algorithms to make them consume lesser energy. As a first step, we try to determine the applicability of the asymptotic energy complexity model by Roy et. al. on these hashes and protocols. Specifically, we try to observe whether parallelizing the access of blocks of data in these algorithms reduces their energy consumption based on the energy model. Our results confirm the applicability of the energy model on these hashes and protocols. Our work is motivated by the relevance and importance of cryptographic hashes and symmetric key protocols for modern ICT (Information and Communication Technology), and ICT enabled industry to keep them protected from dynamically changing threat scenarios. Hence the design of more energy efficient hashes and protocols will definitely contribute in reducing the ICT energy consumption that is continuously increasing.

Keywords: secured hashes; symmetric key encryption; energy optimization

---

1. Introduction

Motivation to consider energy efficiency in delivering information technology solutions comes from: 1. Data centers with strong focus on energy management for server class systems. 2. Personal computing devices such as smartphones, handhelds, and notebooks, which run on batteries and perform a significant amount of computation and data transfer. 3. Telecom providers expecting to invest in equipment that will form an integral part of the global network infrastructure. On the other hand, information security has become a natural component of all technology solutions. Security protocols consuming additional energy are often incorporated in these solutions. Thus, the impact of security protocols on energy consumption needs to be studied. Ongoing research in this context has been mainly focused on energy efficiency/consumption on specific hardware and/or different systems/platforms. Very little is known or has been explored regarding energy consumption or efficiency from an *applications* perspective, although apps for smartphones and handhelds abound.
Our Contributions: Our work makes a few key contributions. We first, present a detailed experimental evaluation of energy consumption for two distinct classed of cryptographic protocols. The first is the hash functions in which we cover MD2, MD5, SHA-1, and SHA-2 hashes. The second is symmetric key encryption in which we cover Blowfish, DES, 3DES, and AES ciphers. Our work experimentally validates the applicability of the energy model proposed in [1] on these classes of cryptographic protocols. We also present a generic way to achieve any desired degree of memory parallelism for a given protocol (hash or symmetric key), which can be employed by software engineers to implement energy efficient cryptographic protocols. We observe that the increased parallelism reduces running time along with energy. Second, we identify the main parameters of the protocols that impact their energy consumption. Our final contribution is a set of practical recommendations, derived from our experimental study, for energy-aware security protocol design.

2. Background

Cryptography has evolved from the earliest forms of secret writing to current era of computationally secure protocols, addressing range of security issues. In modern age, cryptography is not only about encryption, but it has larger objective of ensuring data protection from adversary's activities. Scope of modern cryptography also includes techniques and protocols to achieve authentication, nonrepudiation, and integrity objectives. Complexity of cryptology methods and its applications have continuously increased and evolution of computers has given a completely new dimension to this. Now cryptography problems/algorithms are measured in terms of computational hardness. In this journey, cryptography has always received a threat of getting obsolete because of rapidly increasing computational capabilities.

However, cryptography techniques still have great relevance and importance for modern ICT (Information and Communication Technology), and ICT enabled industry to keep them protected from dynamically changing threat scenarios [2–6].

Energy has become a first class parameter now a days. This has been triggered with the ever-increasing energy generation by the data centers, and with the advent of the hand-held battery-driven devices like laptops, PDAs, etc. Cryptographic protocols have become an integral part of these devices to make them secured. Also, the cryptographic protocols are generally very expensive in terms of their energy consumption. Therefore especially for hand held devices, the protocols drain out a lot of battery power while operating. A network flooding attack with the intention of causing a simple denial of service by depleting the battery life of the device has been illustrated in [7]. They show that these flooding attacks can be carried out utilizing a smartphone as the aggressor in order to attack other mobile devices and that the procedure for such attacks is not difficult. A simple tool has been developed in order to carry out these attacks and to show that even though these attacks are relatively simple, they can have profound effects.

Therefore the necessity of reducing the energy consumption of cryptographic protocols comes into picture. As mentioned, we have considered two classes of cryptographic protocols to implement a parallelism technique based on the energy model of [1] that leads to reduction in their energy consumption.

2.1. Related work

Work has been done to show that hash functions are computationally intensive and are used in mobile devices. In this [8] paper, the authors have analyzed the energy efficiency versus quality characteristics of several hash functions. They perform Avalanche and Chi-square test to analyze the energy usage in Java-enabled smart phone. With the result obtained, the authors propose the best cryptographic and non-cryptographic algorithms that can be used in mobile devices [8].

The impact of various parameters at the protocol level (such as cipher suites, authentication mechanisms, and transaction sizes, etc.) and the cryptographic algorithm level (cipher modes, strength) on overall energy consumption for secure data transactions has been investigated in [9]. Based on their results, the authors discuss various opportunities for realizing energy-efficient implementations of security protocols. Further work on these lines has been done in [10–15].

However, in our opinion, the recommendations proposed above for energy reductions in cryptographic protocols though very nice, stop short of making energy minimization a primary goal of security protocols design. In other words, the works do not address the questions: "(1) How do we design secured cryptographic protocols that are energy optimal? (2) How do we modify existing cryptographic protocols to make them energy optimal?". The main impediment for pursuing this direction is a to prove the applicability of a realistic model of energy consumption at the algorithm design by [1] on some of these protocols. We believe that our work will complement this body of existing work.

3. Engineering Cryptographic Algorithms for Energy Efficiency

3.1. Energy Complexity Model

An asymptotic energy complexity model for algorithms was proposed in [1]. Inspired by the popular DDR3 architecture, the model assumes that the memory is divided into P *banks* $(M_1, M_2, \cdots, M_P)$ each of which can store multiple blocks of size B. Each bank M_i has its own cache C_i that can hold exactly one block. A block can be accessed only when it is in the cache of the bank it belongs to. Therefore only one block of a particular bank can be accessed at a particular time. But P blocks in P *different* memory banks can be accessed in parallel. The main contribution of the model in [1] was to highlight the effect of parallelizability of the memory accesses in energy consumption. In particular, the energy consumption of an algorithm was derived as the weighted sum $T + (PB) \cdot I$, where T is the total time taken and I is the number of *parallel* I/Os made by the algorithm. However, this model measures the energy consumption asymptotically and in particular, ignores constants in order to derive the simple model above.

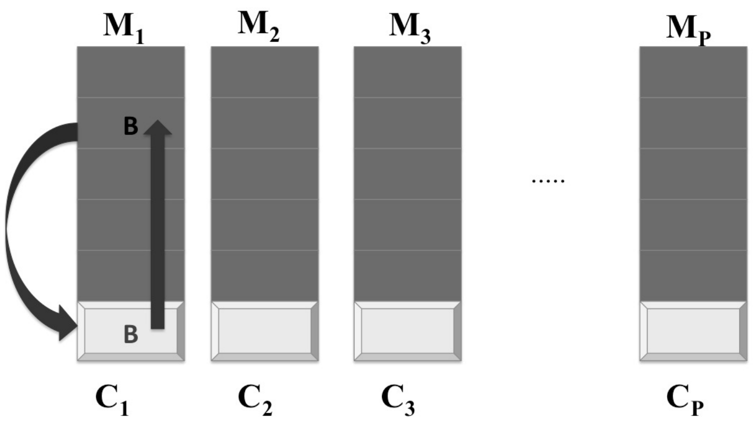


Figure 1. The Energy Complexity Model.

3.2. P-way Parallelism for Blocks

In the both classes of cryptographic algorithms we consider for energy efficiency (hashes and symmetric key), the input (plaintext) is in a vector form, and it is divided into blocks of a given size B, which are then processed. In this section, we describe a generic technique we adopt to parallelize the input blocks to the cryptographic algorithms based on the results in [16].

Given any input data (vector), we created a logical mapping which ensures access to the vector in P-way parallel fashion, where P ranges from 1 to 8. As we mentioned before, the input data (vector) for both the classes of algorithms are accessed block-wise with predefined sized blocks. Consider an input vector $\mathbf{V}$ size of N. Our mapping function logically converts $\mathbf{V}$ into a matrix $\mathbf{M}$ of dimensions $\frac{N}{B} \times B$, where B is the size of each block.

We then define a *page table* vector $\mathbf{T}$ of size $\frac{N}{B}$ that contains the *ordering* of the blocks. The ordering is based on the way we want to access the blocks (P-way would mean a full parallel access). The page table is populated by picking blocks with jumps. For a 1-way access, we select jumps of P ensuring the consecutive blocks are in the same bank. For a P-way access, we selects jumps of 1 i.e. the blocks are picked from banks in round robin order. Fig. 2 presents an example with 4-way and 2-way parallel access.

The division of the input into blocks are done differently for hashes and symmetric key algorithms respectively. The specifics of the mappings for each of them will be discussed when we illustrate each one in detail.

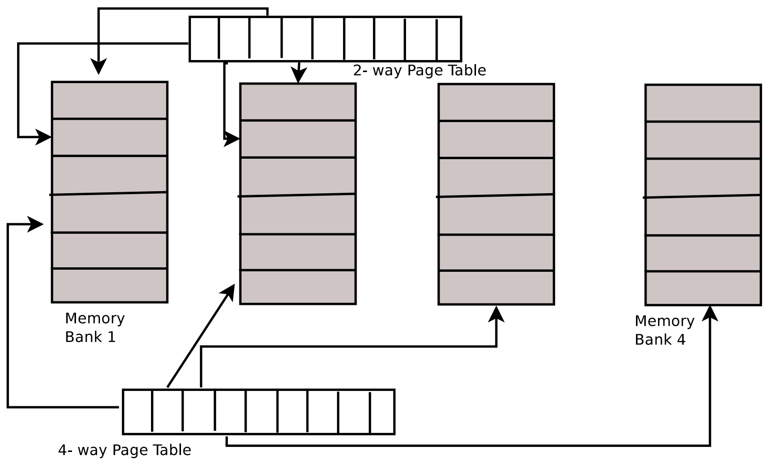


Figure 2. Memory layout for achieving various degree of parallelism for $P = 4$.

```
Input: Page table vector V, jump amount $jump$.
$factor = 0$;
for $i = 0$ to $\frac{N}{B} - 1$ do
    if $i > 1$ and $((i \times jump) \pmod{\frac{N}{B}}) = 0)$ then
        $factor = factor + 1$;
    end
    $\mathbf{V}_i = (i \times jump + factor) \pmod{\frac{N}{B}}$;
end
```

Algorithm 1: The function to create an ordering among the blocks

3.2.1. Code Optimization

As the memory layout scheme is implemented on top of the algorithm, it might lead to additional overhead, negatively impacting the energy savings. Hence we incorporated some optimization techniques to reduce this overhead.

1. **Usage of Spatial Locality.** For every memory access we need to call the map function, which creates an overhead. We used the concept of Spatial Locality (consecutive elements in a logical stride are placed on consecutive memory locations) to reduce the overhead.
2. **Bit shifts.** We replaced operations like multiplication, division and mod with bit shifts wherever possible.
3. **Register variables.** To reduce runtime, we utilized Register variables when a variable had to be computed numerously.

The benchmark code was written in C and was compiled using `gcc` compiler.

4. Experimental Setup and Methodology

The two sets of cryptographic protocols we chose to experiment on are hashes and symmetric key encryption protocols.

4.1. Protocols Studied

We considered a mix of hashes and symmetric key protocols. We experimented on a few old ones, and the most recent ones. Our experiments are primarily designed to test whether the energy model of [1] is applicable to these protocols. In other words, can we observe a variation in the energy consumption of these protocols by varying the *degree of parallelism* of their inputs. Hence,

Table 1. The hash functions considered.

Hash	Output Size (bits)	Vulnerabilities
MD2	128	Preimage and collision
MD5	128	2^{64} complexity collision
SHA-1	160	2^{80} complexity collision, 2^{61} hard to implement
SHA-2	256	None

we implement variants of the protocols that allow us to use 1, 2, 4, or 8 memory banks in parallel on DDR3 memory with 8 memory banks.

Below we list the benchmarks used in this study.

1. **Hashes**: The hashes we consider are MD2 [17], MD5 [18], SHA-128 [19], and SHA-256 [20].
2. **Symmetric Key Protocols**: The protocols we consider are Blowfish [21], DES [22–24], 3-DES (triple DES) [25], and AES [26–29].

Each experiment has been repeated 10 times and the means are reported in this work. The variations in the energy readings that is, deviations from the average values were small (less than 4%) for all the experiments performed.

4.2. Hardware Setup

The experiments were run on a Mac note book with 4 Intel cores, each of which operates at a frequency of 2 GHz. The laptop has 4GB DDR3 memory with 8 banks and is running Mac OS X Lion 10.7.3. The sizes of L2 Cache (per Core), and L3 Cache are respectively 256 KB and 6 MB. The disk size of the machine is 499.25 GB. During any experimental run, all non-essential processes were aborted to ensure that the power measured by us can be attributed primarily to the application being tested. We measured the (total) power drawn using the Hardware Monitor tool [30] which include the processor power, memory (RAM) power, and a background (leakage) power. The Hardware Monitor tool reads sensor data directly from Apple's System Management Controller (SMC), which is an auxiliary processor independent of the main computer. Hence, energy consumed by the monitoring infrastructure does not add to the sensor readings, ensuring that the reported measurements are fairly accurate.

5. The hash functions

Table 1 lists the hashes considered along with the sizes of their output, and any known vulnerabilities.

The output sizes specifications suggest, the output is always of that size irrespective of the size of the input.

5.1. P-way Parallelism for hashes

Energy optimal algorithms proposed in [1] require data to be laid out in memory with a controlled degree of parallelism. We first propose a way to ensure desired memory parallelism for a given input M to the hash algorithm.

5.1.1. Merkle-Damgård construction

All of the hashes we consider (MD2, MD5, SHA-1, and SHA-2) are based on the Merkle-Damgård construction [31–33]. In this construction, The given input M is *padded* with a few bits at the end to

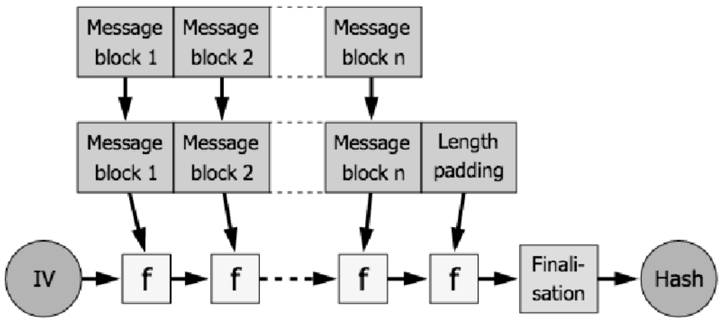


Figure 3. Merkle-Damgård construction.

make it a multiple of a chosen block size B. The hash is then produced by applying a chosen function f on each block, and combining the result.

As described in Fig. 3, the algorithm starts with an initial value, the initialization vector (IV). The IV is a fixed value (algorithm or implementation specific). For each message block, the compression (or compacting) function f takes the result so far, combines it with the message block, and produces an intermediate result. The last block is padded with zeros as needed and bits representing the length of the entire message are appended.

5.2. Implementing the parallelism

For the message M, which is a multiple of blocks of B bytes, we created a logical mapping which ensures access to the blocks in P-way parallel fashion, where P ranges from 1 to 8. More specifically, when $P = 1$, (almost) all the blocks of B bytes are clustered in a single bank. While for $P = 8$, the blocks are evenly spread across all 8 banks to ensure the maximum degree of parallelism of the access to the input (M). We also experiment for $P = 2$, and $P = 4$.

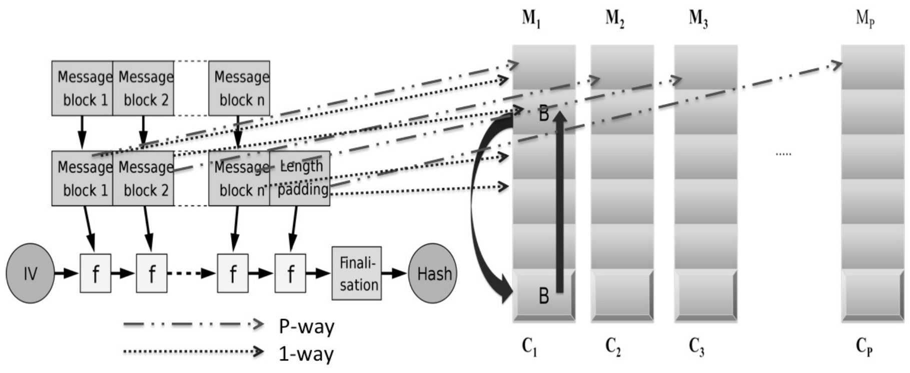


Figure 4. Implementing parallelism for hashes.

To achieve the above, we create a mapping function which maps the physical input M into the logical input which defines the degree of parallelism (Fig. 6). In other words, we define an *ordering* among the blocks of B bytes which defines the logical input (and the degree of parallelism). We change the *order of processing of the blocks* to ensure the change in parallelism of the input M. So e.g. for 1-way parallelism, every eight consecutive block access would be in the same bank; whereas for 8-way parallelism, every eight consecutive block accesses would be spread across the 8 different banks.

Table 2. The symmetric key protocols considered.

Hash	Input Block Size (bits)	Key Size (bits)	Vulnerabilities
Blowfish	64	32–448	Weakness in key selection
DES	64	56	Bruteforcing over keys, Cryptanalysis
3-DES	64	56, 112, or 168	Computationally inefficient
AES	128	128, 192, or 256	None

6. Symmetric Key Encryption Protocols

Table 2 lists the symmetric key protocols considered along with the sizes of their input blocks, keys, and any known vulnerabilities.

6.1. *P-way Parallelism for the symmetric key protocols*

6.1.1. Block ciphers

Each of the protocols considered (Blowfish, DES, 3-DES, and AES) are *block ciphers* operating on fixed-length groups of bits, called blocks (of size B), with an unvarying transformation that is specified by a symmetric key.

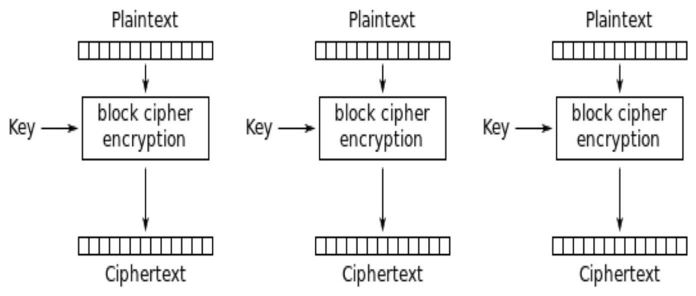


Figure 5. Block ciphers.

Fig. 5 how each of the protocols (block ciphers) work.

6.2. *Implementing the parallelism*

We treat each symmetric key algorithm (protocol) as a black box. Given an input M, the protocol divides M into blocks of B bytes and processes each block for encryption to produce the ciphertext (Figure 5). For the message M which is a multiple of blocks of B bytes, we created a logical mapping which ensures access to the blocks in P-way parallel fashion, where P ranges from 1 to 8. More specifically, when $P = 1$, (almost) all the blocks of B bytes are clustered in a single bank. While for $P = 8$, the blocks are evenly spread across all 8 banks to ensure the maximum degree of parallelism of the access to the input (M). We also experiment for $P = 2$, and $P = 4$.

To achieve the above, we create a mapping function which maps the physical input M into the logical input which defines the degree of parallelism. In other words, we define an *ordering* among the blocks of B bytes which defines the logical input (and the degree of parallelism). We change the *order of processing of the blocks* to ensure the change in parallelism of the input M.

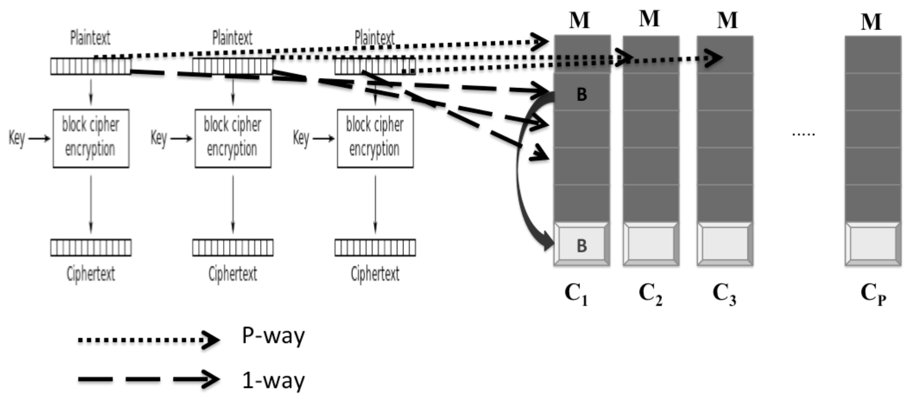


Figure 6. Implementing parallelism for symmetric key protocols.

7. Experimental Results

7.1. Effect of Parallelism on Hash functions

In our first set of experiments, we compare the hash functions MD2 and MD5. We observe (Fig. 7(a) and (b)) that memory parallelism has significant impact on energy consumption for MD2 and MD5. That is, we observe that energy consumption of these hashes decreases with the increase in the degree of parallelism of the blocks. This is in line with energy model proposed in [1]. Interestingly the MD5 hash shows more significant reductions in energy consumption for different degrees of parallelism.

Next we compare the hash functions SHA-1 and SHA-2. Fig. 8(a) and (b) show a similar trend in the energy consumption as before that is again along the expected lines with the energy model of [1].

7.2. Effect of Parallelism on Symmetric key encryption functions

For the symmetric key protocols, we first compare the DES and 3DES. We observe (Fig. 9(a) and (b)) that memory parallelism again has significant impact on energy consumption for DES and 3DES. Also, as expected 3DES consumes about thrice the amount of energy DES consumes.

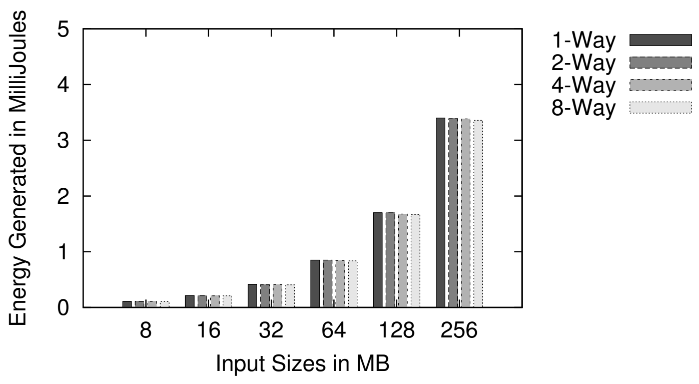
Next we compare the symmetric key protocols AES and Blowfish. Fig. 10(a) and (b) show a similar trend in the energy consumption as before that is again along the expected lines with the energy model of [1]. We use a 256 bit key for AES in the experiments.

For the symmetric key protocols in general the variations in energy consumption is bigger for larger size inputs (e.g. 256 MB). Whereas we see little variations for small inputs (e.g. 8MB). This is due to the fact that small size inputs fit into the caches nullifying the effect of memory parallelism in the RAM on the energy consumption. Also, in general, the energy consumption variations seem to work better for the symmetric key protocols than on the hash functions.

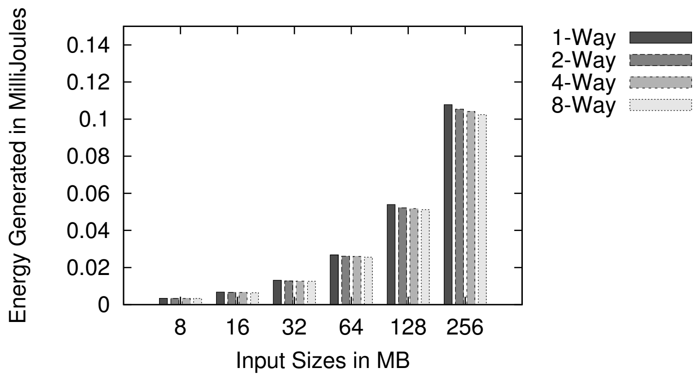
7.3. Comparison of energy consumed

We also compare the energy consumed by various hash functions, and symmetric key protocols.

As we see from Fig. 11(a), MD2 consumes way more energy than the other three, which consumes comparable amount of energy. For the symmetric protocols, (Fig. 11(b)), AES consumes more energy than 3DES because of the fact we work with a 256 bit key for AES as compared to a 168 bit key for 3DES. The interesting observation is how light Blowfish is in terms of energy consumption.

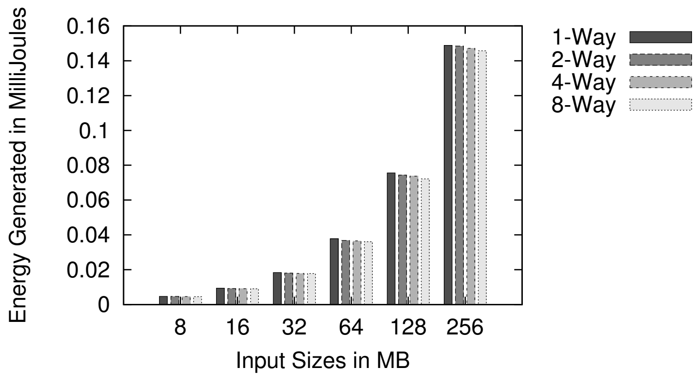


(a)

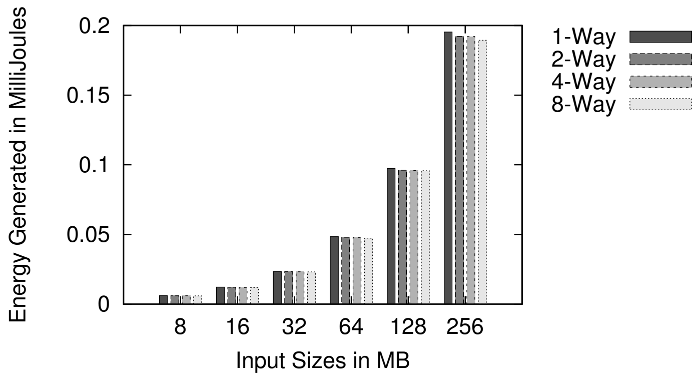


(b)

Figure 7. Energy Consumption by (a) MD2 and (b) MD5 for various input sizes on different degrees of parallelism.

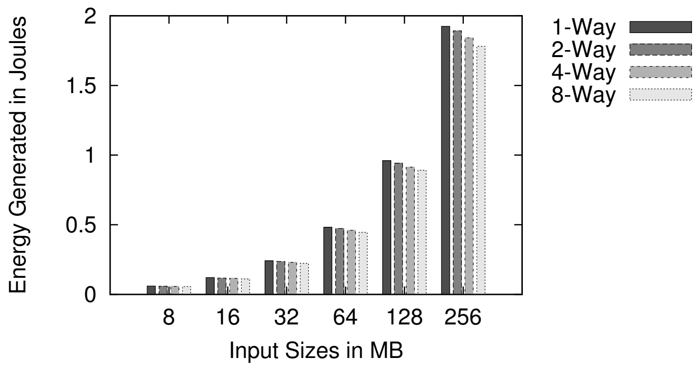


(a)

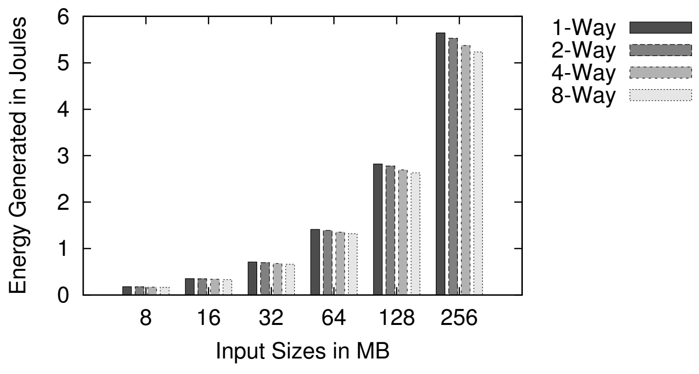


(b)

Figure 8. Energy Consumption by (a) SHA-1 and (b) SHA-2 for various input sizes on different degrees of parallelism.

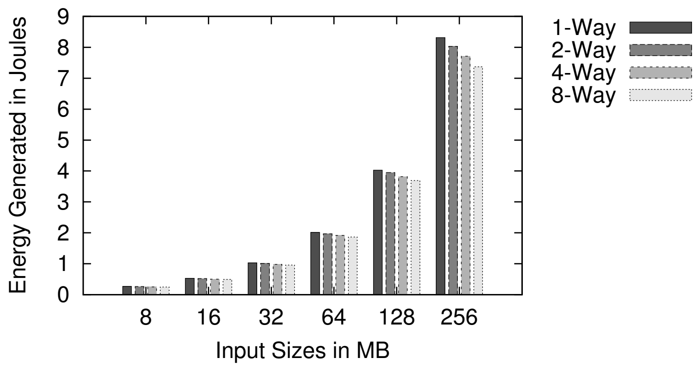


(a)

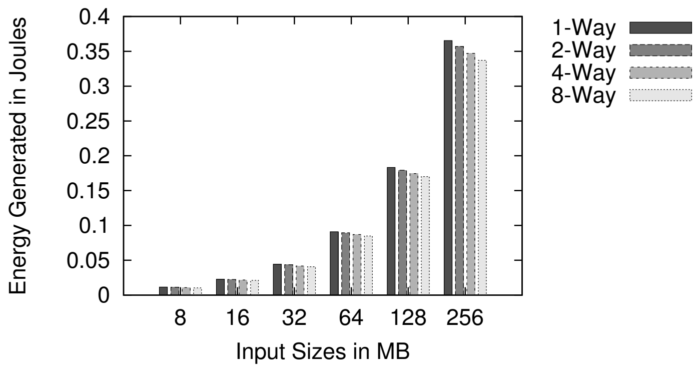


(b)

Figure 9. Energy Consumption by (a) DES and (b) Triple-DES for various input sizes on different degrees of parallelism.

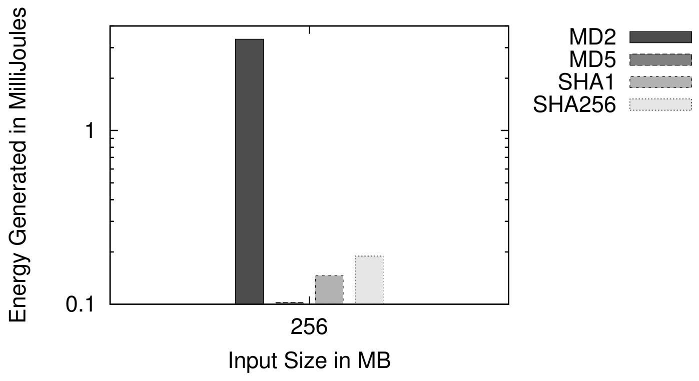


(a)

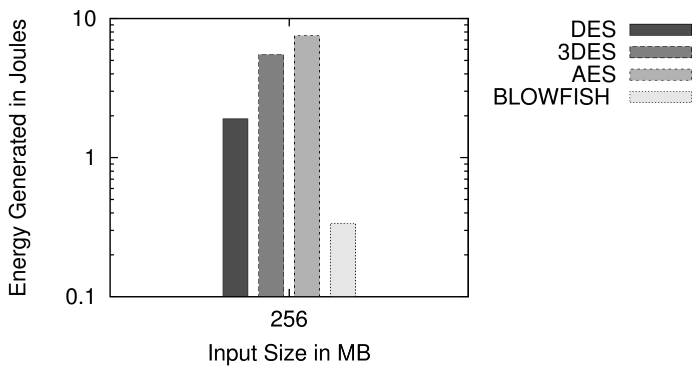


(b)

Figure 10. Energy Consumption by (a) AES and (b) Blowfish for various input sizes on different degrees of parallelism.



(a)



(b)

Figure 11. Energy Consumption by (a) Hashes and (b) Symmetric key protocols for a fixed input size.

8. Analysis of Results

8.1. Applicability of the energy complexity model

Our experimental study establishes variations in energy consumed for both the classes of cryptographic algorithms with the variations in parallelization of their block accesses. Overall, these results establish the applicability of the energy complexity model proposed in [1] on these classes of cryptographic algorithms . One of the key aspects highlighted by our experiments is the reduced impact of parallelism, when the input data is small. This was due to the fact that smaller inputs fit into the cache. The memory controller then applies smart techniques to perform write backs to make the accesses more efficient (esp. during low parallelism), and in the process negates the effect of parallelism.

8.2. Analysis for hashes

With the applicability of the energy model established we further conducted micro experiments to compare the energy consumption of SHA-2 during our best case of parallelism, (8-way) with that during we have no parallelism at all (Fig. 12).

We see differences in energy consumption, but the difference at the moment is not very significant. However, as before, the difference becomes more significant with larger inputs. Hence, we can conclude that if the data is large, using the 8-way parallel technique is more energy efficient than not using any parallelism at all (just the regular implementation).

Recommendations for hash usage. Fig. 11(a) shows MD5 to be the most efficient in terms of energy consumption. But from Table 1, we know MD5 is vulnerable to the collision attack, whereas no vulnerabilities for SHA-2 are currently known. To further investigate if MD5 might be still usable to conserve energy in some places (and it is still in use), we estimated the numbers in Table 3.

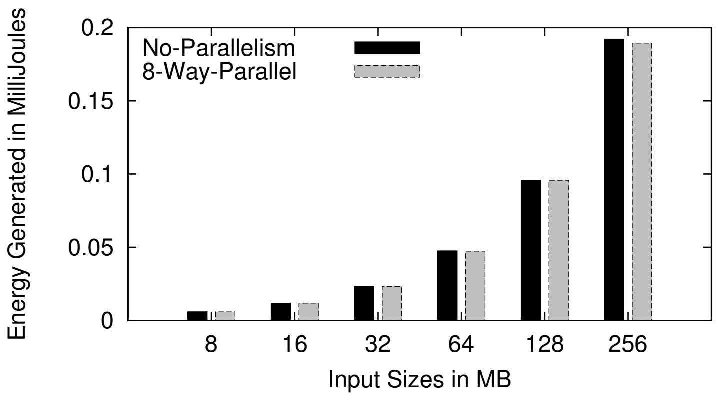


Figure 12. Energy Consumption of SHA-2 with no parallelism and 8-way parallelism.

Table 3. Time required to perform collision attacks.

Hash	**Vulnerabilities**	**Time Required at 10^9 comparisons/s**	**Time Required at 10^{13} comparisons/s**
MD5	2^{64} complexity collision	2^{64}ns $\approx$ 584.5 years	6 months
SHA-1	2^{80} complexity collision	2^{80}ns $\approx 3.83 \times 10^7$ years	38000 years
SHA-2	None	–	–

What we would like to conclude from the numbers is, the reason why MD5 is still in use is it is not very likely for any adversary to have a supercomputer of the capacity to perform the collision attack on it in a reasonable amount of time (based on the numbers we get). Hence we recommend the use of MD5 over SHA-2 to conserve energy if the user is mostly sure of the strength of the probable adversary.

8.3. Analysis for symmetric key protocols

In a similar way, we further conducted micro experiments to compare the energy consumption of AES during our best case of parallelism, (8-way) with that during we have no parallelism at all(Fig. 13).

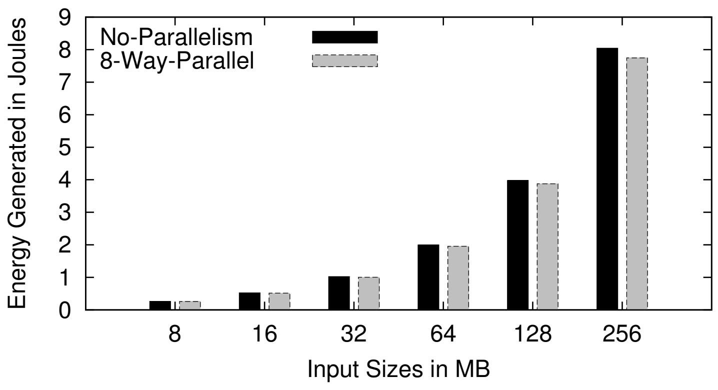


Figure 13. Energy Consumption of AES with no parallelism and 8-way parallelism.

Again, we see differences in energy consumption, but the difference at the moment is not very significant. However, as before, the difference becomes more significant with larger inputs. Hence, we can conclude that if the data is large, using the 8-way parallel technique is more energy efficient than not using any parallelism at all (just the regular implementation).
Recommendations for symmetric key protocols usage. Fig. 11(b) shows Blowfish to be the most efficient in terms of energy consumption. But Blowfish becomes computationally intense when it comes to key replacement. We have not conducted how expensive is the key replacement for Blowfish in terms of energy consumption. We would recommend the usage of Blowfish over the other protocols if the user is not required to change the keys too often.

8.4. Hashes vs. symmetric key protocols

There is a fundamental difference in how we parallelized the hashes and the symmetric key protocols. In case of the hashes, the parallelization of the block is done *inside* the hash function in the phase of the Merkle-Damgård construction, where the blocks are formed and accessed. Whereas in case of the symmetric key protocols, we do the parallelization of the blocks *before* they are applied for encryption (as a preprocessing phase). Comparing the results of Fig. 7(a) & (b) and Fig. 8(a) & (b) with Fig. 9(a) & (b) and Fig. 10(a) & (b), we observe, the parallelization for the symmetric key protocols appear to be more effective. In other words, the symmetric key protocols exhibit larger variations in energy consumption than the hashes by applying parallelization.

9. Conclusion

To summarize, we have proved the applicability of the energy model by [1] on two important classes of cryptographic protocols, which is our main result. Additionally, we have discovered a few things that might be instrumental towards designing energy aware secured cryptographic protocols. A few questions worth pursuing are:

1. Can we develop more efficient techniques to parallelize block accessed in hashed and symmetric key protocols?

   Using Meta-Blocks. To throw some light on this point, the machine we used had a version of DDR3 RAM where the banks had a block size of 256 bits. Therefore, a single block of DES (size of 64 bits) AES (size of 128 bits) does not entirely fill up a single block in the memory banks. We define a *meta-block* to be formed of one or more blocks. Therefore, for example in the case of AES, we could work with meta-blocks of size 256 bits, where each meta-block would contain two AES blocks. We believe this additional level of abstraction to help further reduce the energy consumption in symmetric key protocols.
2. Is the energy model by [1] applicable on public key protocols?

   Public key protocols depend on the computational hardness of one-way functions like modular-exponentiation or elliptic curve cryptography (ECC). It will be interesting to investigate whether we can parallelize computations for those operations, and whether parallelization reduces energy consumption in public key protocols like RSA or Diffie-Helman.
3. We have only considered encryptions for the symmetric key protocols. Do the decryption techniques for these protocols show the similar results?

   Decryption process in some symmetric-key ciphers (e.g. DES) is exactly the same operation(s) as encryption. But ciphers like AES have different decryption operation(s) than encryption. We have applied the energy model to the encryption processes of the symmetric-key ciphers, our belief is the model will be applicable to the decryption processes as well, and we expect to observe similar results.

Bibliography

1. Roy, S.; Rudra, A.; Verma, A. An energy complexity model for algorithms. Proceedings of the 4th conference on Innovations in Theoretical Computer Science. ACM, 2013, pp. 283–304.
2. Zhang, X.; Du, H.t.; Chen, J.q.; Lin, Y.; Zeng, L.j. Ensure data security in cloud storage. Network Computing and Information Security (NCIS), 2011 International Conference on. IEEE, 2011, Vol. 1, pp. 284–287.
3. Mishra, M. Improved Cloud Security Approach with Threshold Cryptography **2015**.
4. Preneel, B. Cryptography and information security in the post-Snowden era. Proceedings of the First International Workshop on TEchnical and LEgal aspects of data pRIvacy. IEEE Press, 2015, pp. 1–1.
5. Yang, J.S.; Lee, H.J.; Park, M.W.; Eom, J.h. Security Threats on National Defense ICT based on IoT **2015**.
6. Sheng, S.; Wu, X. A new digital anti-counterfeiting scheme based on chaotic cryptography. ICT Convergence (ICTC), 2012 International Conference on. IEEE, 2012, pp. 687–691.
7. Salerno, S.; Sanzgiri, A.; Upadhyaya, S. Exploration of attacks on current generation smartphones. *Procedia Computer Science* **2011**, *5*, 546–553.
8. Damasevicius, R.; Ziberkas, G.; Stuikys, V.; Toldinas, J. Energy Consumption of Hash Functions. *Elektronika ir Elektrotechnika* **2012**, *18*, 81–84.
9. Potlapally, N.R.; Ravi, S.; Raghunathan, A.; Jha, N.K. Analyzing the energy consumption of security protocols. Proceedings of the 2003 international symposium on Low power electronics and design. ACM, 2003, pp. 30–35.
10. Toldinas, J.; Štuikys, V.; Ziberkas, G.; Naunikas, D. Power awareness experiment for crypto service-based algorithms. *Elektronika ir Elektrotechnika* **2015**, *101*, 57–62.
11. Potlapally, N.R.; Ravi, S.; Raghunathan, A.; Jha, N.K. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *Mobile Computing, IEEE Transactions on* **2006**, *5*, 128–143.
12. Elminaam, D.S.A.; Abdual-Kader, H.M.; Hadhoud, M.M. Evaluating The Performance of Symmetric Encryption Algorithms. *IJ Network Security* **2010**, *10*, 216–222.
13. Prasithsangaree, P.; Krishnamurthy, P. Analysis of energy consumption of RC4 and AES algorithms in wireless LANs. Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE. IEEE, 2003, Vol. 3, pp. 1445–1449.
14. Kaps, J.P.; Sunar, B. Energy comparison of AES and SHA-1 for ubiquitous computing. In *Emerging directions in embedded and ubiquitous computing*; Springer, 2006; pp. 372–381.
15. Nie, T.; Song, C.; Zhi, X. Performance evaluation of DES and Blowfish algorithms. Biomedical Engineering and Computer Science (ICBECS), 2010 International Conference on. IEEE, 2010, pp. 1–4.
16. Roy, S.; Rudra, A.; Verma, A. Energy Aware Algorithmic Engineering. Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on. IEEE, 2014, pp. 321–330.
17. Kaliski, B. The MD2 message-digest algorithm **1992**.
18. Rivest, R. The MD5 message-digest algorithm **1992**.
19. Eastlake 3rd, D.; Jones, P. US secure hash algorithm 1 (SHA1). Technical report, 2001.
20. Sklavos, N.; Koufopavlou, O. On the hardware implementations of the SHA-2 (256, 384, 512) hash functions. Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on. IEEE, 2003, Vol. 5, pp. V–153.
21. Schneier, B. The Blowfish encryption algorithm. *Dr Dobb's Journal-Software Tools for the Professional Programmer* **1994**, *19*, 38–43.
22. Standard, D.E. FIPS pub 46. *Appendix A, Federal Information Processing Standards Publication* **1977**.
23. Sklower, K.; Meyer, G.M. The PPP DES Encryption Protocol, Version 2 (DESE-bis) **1998**.
24. EOZ, P. Data Encryption Standard. *Computers and Security*, *7*.
25. Kummert, H. The PPP Triple-DES Encryption Protocol (3DESE) **1998**.
26. Miller, F.P.; Vandome, A.F.; McBrewster, J. Advanced Encryption Standard **2009**.
27. Daemen, J.; Rijmen, V. *The design of Rijndael: AES-the advanced encryption standard*; Springer Science & Business Media, 2013.
28. Selent, D. Advanced encryption standard. *Rivier Academic Journal* **2010**, *6*, 1–14.
29. Rijmen, V.; Daemen, J. Advanced encryption standard. *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology* **2001**, pp. 19–22.

30. Software-Systeme, M.B. Hardware Monitor. http://www.bresink.com/osx/HardwareMonitor.html. Accessed: 2015-07-01.
31. Damgård, I.B. A design principle for hash functions. Advances in Cryptology—CRYPTO'89 Proceedings. Springer, 1990, pp. 416–427.
32. Merkle, R.C. Secrecy, authentication, and public key systems. **1979**.
33. Merkle, R.C. A certified digital signature. Advances in Cryptology—CRYPTO'89 Proceedings. Springer, 1990, pp. 218–238.